Team : 70

Magazine Management System

TA : Marwa Helaly

2021170393	فرح معتز محمد فؤاد عباس
2021170439	مايا محمد محمد حلمى الطيب
2021170130	بيشوى سدرة صابر سدرة
2021170558	مهند خالد على محمد
2021170418	مؤمن محمد احمد قدرى محمد
2021170590	نور الدين هشام محمد نور

Department : IS

```sql
--------------------
------ TABLES ------
--------------------

-- Create USER table
CREATE TABLE USERS (
    id NUMBER,
    email VARCHAR2(255) UNIQUE,
    username VARCHAR2(100),
    password VARCHAR2(255),
    role VARCHAR2(20) CHECK (role IN ('ADMIN', 'AUTHOR', 'READER')),
    PRIMARY KEY (id)
);

-- Create CATEGORY table
CREATE TABLE CATEGORIES (
    id NUMBER,
    name VARCHAR2(100),
    PRIMARY KEY (id)
);

-- Create MAGAZINE table
CREATE TABLE MAGAZINES (
    id NUMBER,
    name VARCHAR2(255),
    admin_id NUMBER,
    publication_date DATE,
    PRIMARY KEY (id),
    FOREIGN KEY (admin_id) REFERENCES USERS(id)
);


-- Create ARTICLE table
CREATE TABLE ARTICLES (
    id NUMBER,
    title VARCHAR2(255),
    magazine_id NUMBER,
    author_id NUMBER,
    category_id NUMBER,
    content VARCHAR2(2000),
    status VARCHAR2(20) CHECK (status IN ('PENDING', 'APPROVED', 'REJECTED')),
    PRIMARY KEY (id),
    FOREIGN KEY (magazine_id) REFERENCES MAGAZINES(id),
    FOREIGN KEY (author_id) REFERENCES USERS(id),
    FOREIGN KEY (category_id) REFERENCES CATEGORIES(id)
);


-- Create MAGAZINE_ARTICLES table
CREATE TABLE MAGAZINE_ARTICLES (
    magazine_id NUMBER,
    article_id NUMBER,
    PRIMARY KEY (magazine_id, article_id),
    FOREIGN KEY (magazine_id) REFERENCES MAGAZINES(id),
    FOREIGN KEY (article_id) REFERENCES ARTICLES(id)
);
```

```
-- Create SUBSCRIPTION table
CREATE TABLE SUBSCRIPTIONS (
    reader_id NUMBER,
    magazine_id NUMBER,
    start_date DATE,
    end_date DATE,
    price NUMBER,
    PRIMARY KEY (reader_id, magazine_id),
    FOREIGN KEY (reader_id) REFERENCES USERS(id),
    FOREIGN KEY (magazine_id) REFERENCES MAGAZINES(id)
);
```

```
------------------------------
--------- SEQUENCES ----------
------------------------------

-- Create sequences
CREATE SEQUENCE user_seq START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE category_seq START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE article_seq START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE magazine_seq START WITH 1 INCREMENT BY 1;
```

```
--------------------------
-------- TRIGGERS --------
--------------------------

-- Trigger for USERS table
CREATE OR REPLACE TRIGGER user_trigger
BEFORE INSERT ON USERS
FOR EACH ROW
BEGIN
   SELECT user_seq.nextval INTO :new.id FROM dual;
END;
/

-- Trigger for CATEGORY table
CREATE OR REPLACE TRIGGER category_trigger
BEFORE INSERT ON CATEGORIES
FOR EACH ROW
BEGIN
   SELECT category_seq.nextval INTO :new.id FROM dual;
END;
/

-- Trigger for ARTICLE table
CREATE OR REPLACE TRIGGER article_trigger
BEFORE INSERT ON ARTICLES
FOR EACH ROW
BEGIN
   SELECT article_seq.nextval INTO :new.id FROM dual;
```

```
END;
/

-- Trigger for MAGAZINE table
CREATE OR REPLACE TRIGGER magazine_trigger
BEFORE INSERT ON MAGAZINES
FOR EACH ROW
BEGIN
   SELECT magazine_seq.nextval INTO :new.id FROM dual;
END;
/
```

```
-----------------------------
---- MAGAZINE PROCEDURES ----
-----------------------------

-- Create procedure for inserting a new magazine
CREATE OR REPLACE PROCEDURE insert_magazine(
   name_in IN VARCHAR2,
   admin_id_in IN NUMBER,
   publication_date_in IN DATE
) AS
BEGIN
   INSERT INTO MAGAZINES (name, admin_id, publication_date)
   VALUES (name_in, admin_id_in, publication_date_in);
```

```
    COMMIT;
END insert_magazine;
/

-- Create procedure for updating magazine information
CREATE OR REPLACE PROCEDURE update_magazine(
    id_in IN NUMBER,
    name_in IN VARCHAR2,
    admin_id_in IN NUMBER
) AS
BEGIN
    UPDATE MAGAZINES
    SET name = name_in,
        admin_id = admin_id_in
    WHERE id = id_in;
    COMMIT;
END update_magazine;
/

-- Create procedure for deleting a magazine
CREATE OR REPLACE PROCEDURE delete_magazine(
    id_in IN NUMBER
) AS
BEGIN
    DELETE FROM MAGAZINES
    WHERE id = id_in;
    COMMIT;
END delete_magazine;
/
```

```
-- Create procedure for retrieving magazine information
CREATE OR REPLACE PROCEDURE get_magazine(
   id_in IN NUMBER,
   magazine_out OUT SYS_REFCURSOR
) AS
BEGIN
   OPEN magazine_out FOR
   SELECT * FROM MAGAZINES
   WHERE id = id_in;
END get_magazine;
/

CREATE OR REPLACE PROCEDURE get_all_magazines(
   magazine_cursor OUT SYS_REFCURSOR
)
AS
BEGIN
   OPEN magazine_cursor FOR
      SELECT *
      FROM MAGAZINES;
END get_all_magazines;
/
CREATE OR REPLACE PROCEDURE get_all_other_magazines (
   p_reader_id IN NUMBER,
   magazines_cursor OUT SYS_REFCURSOR
) AS
BEGIN
   OPEN magazines_cursor FOR
```

```
        SELECT ID, NAME, ADMIN_ID, PUBLICATION_DATE
        FROM MAGAZINES
        WHERE ID NOT IN (
            SELECT MAGAZINE_ID
            FROM SUBSCRIPTIONS
            WHERE READER_ID = p_reader_id
        );
END get_all_other_magazines;
/

CREATE OR REPLACE PROCEDURE get_all_my_magazines (
    p_reader_id IN NUMBER,
    magazines_cursor OUT SYS_REFCURSOR
) AS
BEGIN
    OPEN magazines_cursor FOR
        SELECT ID, NAME, ADMIN_ID, PUBLICATION_DATE
        FROM MAGAZINES
        WHERE ID IN (
            SELECT MAGAZINE_ID
            FROM SUBSCRIPTIONS
            WHERE READER_ID = p_reader_id
        );
END get_all_my_magazines;
/


CREATE OR REPLACE PROCEDURE get_all_my_magazines_articles (
    p_reader_id IN NUMBER,
    articles_cursor OUT SYS_REFCURSOR
) AS
BEGIN
    OPEN articles_cursor FOR
        SELECT a.ID, a.TITLE, a.MAGAZINE_ID, a.CONTENT
        FROM ARTICLES a
        WHERE a.MAGAZINE_ID IN (
            SELECT s.MAGAZINE_ID
            FROM SUBSCRIPTIONS s
            WHERE s.READER_ID = p_reader_id
        ) AND a.STATUS != 'PENDING'; -- corrected the condition here
END get_all_my_magazines_articles;
/
```

```
-----------------------------------
------- CATEGORY PROCEDURES -------
-----------------------------------



-- Create procedure for inserting a new category
CREATE OR REPLACE PROCEDURE insert_category(
    name_in IN VARCHAR2
) AS
BEGIN
    INSERT INTO CATEGORIES (id, name)
    VALUES (category_seq.nextval, name_in);
    COMMIT;
```

```
END insert_category;
/

CREATE OR REPLACE PROCEDURE get_all_categories(
    category_cursor OUT SYS_REFCURSOR
)
AS
BEGIN
    OPEN category_cursor FOR
        SELECT Id, Name
        FROM categories;
END get_all_categories;
/

-- Create procedure for updating category information
CREATE OR REPLACE PROCEDURE update_category(
    id_in IN NUMBER,
    name_in IN VARCHAR2
) AS
BEGIN
    UPDATE CATEGORIES
    SET name = name_in
    WHERE id = id_in;
    COMMIT;
END update_category;
/

-- Create procedure for deleting a category
CREATE OR REPLACE PROCEDURE delete_category(
    id_in IN NUMBER
) AS
BEGIN
    DELETE FROM CATEGORIES
    WHERE id = id_in;
    COMMIT;
END delete_category;
/

-- Create procedure for retrieving category information
CREATE OR REPLACE PROCEDURE get_category(
    id_in IN NUMBER,
    category_out OUT SYS_REFCURSOR
) AS
BEGIN
    OPEN category_out FOR
    SELECT * FROM CATEGORIES
    WHERE id = id_in;
END get_category;
/
```

```
----------------------------------
----- SUBSCRIPTION PROCEDURES -----
----------------------------------

-- Create procedure for inserting a new subscription
CREATE OR REPLACE PROCEDURE insert_subscription(
    reader_id_in IN NUMBER,
    magazine_id_in IN NUMBER,
    start_date_in IN DATE,
    end_date_in IN DATE,
    price_in IN NUMBER
) AS
BEGIN
    INSERT INTO SUBSCRIPTIONS (reader_id, magazine_id, start_date, end_date, price)
    VALUES (reader_id_in, magazine_id_in, start_date_in, end_date_in, price_in);
    COMMIT;
END insert_subscription;
/

-- Create procedure for updating subscription information
CREATE OR REPLACE PROCEDURE update_subscription(
    reader_id_in IN NUMBER,
    magazine_id_in IN NUMBER,
    start_date_in IN DATE,
    end_date_in IN DATE,
    price_in IN NUMBER
) AS
BEGIN
    UPDATE SUBSCRIPTIONS
    SET start_date = start_date_in,
        end_date = end_date_in,
        price = price_in
    WHERE reader_id = reader_id_in AND magazine_id = magazine_id_in;
    COMMIT;
END update_subscription;
/
```

```
CREATE OR REPLACE PROCEDURE get_subscribed_magazines(
    reader_id_in IN NUMBER,
    magazine_cursor OUT SYS_REFCURSOR
)
AS
BEGIN
    OPEN magazine_cursor FOR
        SELECT m.Id, m.Name, m.Admin_Id, m.Publication_Date
        FROM Subscriptions s
        JOIN Magazines m ON s.Magazine_Id = m.Id
        WHERE s.Reader_Id = reader_id_in;
END get_subscribed_magazines;
/

CREATE OR REPLACE PROCEDURE get_not_subscribed_magazines(
    reader_id_in IN NUMBER,
    magazine_cursor OUT SYS_REFCURSOR
)
AS
BEGIN
    OPEN magazine_cursor FOR
        SELECT Id, Name, Admin_Id, Publication_Date
        FROM Magazines
        WHERE Id NOT IN (
            SELECT Magazine_Id
            FROM Subscriptions
            WHERE Reader_Id = reader_id_in
        );
```

```
END get_not_subscribed_magazines;
/

CREATE OR REPLACE PROCEDURE get_subscribed_articles(
   reader_id_in IN NUMBER,
   article_cursor OUT SYS_REFCURSOR
)
AS
BEGIN
   OPEN article_cursor FOR
      SELECT a.Id, a.Title, a.Content, a.Magazine_Id, a.Author_Id
      FROM Articles a
      JOIN Subscriptions s ON a.Magazine_Id = s.Magazine_Id
      WHERE s.Reader_Id = reader_id_in;
END get_subscribed_articles;
/
-- Create procedure for deleting a subscription
CREATE OR REPLACE PROCEDURE delete_subscription(
   reader_id_in IN NUMBER,
   magazine_id_in IN NUMBER
) AS
BEGIN
   DELETE FROM SUBSCRIPTIONS
   WHERE reader_id = reader_id_in AND magazine_id = magazine_id_in;
   COMMIT;
END delete_subscription;
/
-- Create procedure for retrieving subscription information
CREATE OR REPLACE PROCEDURE get_subscription(
   reader_id_in IN NUMBER,
   magazine_id_in IN NUMBER,
   subscription_out OUT SYS_REFCURSOR
) AS
BEGIN
   OPEN subscription_out FOR
   SELECT * FROM SUBSCRIPTIONS
   WHERE reader_id = reader_id_in AND magazine_id = magazine_id_in;
END get_subscription;
/
-------------------------------
------- USER PROCEDURES -------
-------------------------------

-- Create procedure for inserting a new user
CREATE OR REPLACE PROCEDURE insert_user(
   email_in IN VARCHAR2,
   username_in IN VARCHAR2,
   password_in IN VARCHAR2,
   role_in IN VARCHAR2
) AS
BEGIN
   INSERT INTO USERS (email, username, password, role)
   VALUES (email_in, username_in, password_in, role_in);
   COMMIT;
END insert_user;
/

-- Create procedure for updating user information
```

```
CREATE OR REPLACE PROCEDURE update_user(
   id_in IN NUMBER,
   email_in IN VARCHAR2,
   username_in IN VARCHAR2,
   password_in IN VARCHAR2,
   role_in IN VARCHAR2
) AS
BEGIN
   UPDATE USERS
   SET email = email_in,
      username = username_in,
      password = password_in,
      role = role_in
   WHERE id = id_in;
   COMMIT;
END update_user;
/
-- Create procedure for deleting a user
CREATE OR REPLACE PROCEDURE delete_user(
   id_in IN NUMBER
) AS
BEGIN
   DELETE FROM USERS
   WHERE id = id_in;
   COMMIT;
END delete_user;
/
-- Create procedure for retrieving user information
CREATE OR REPLACE PROCEDURE get_user(
   id_in IN NUMBER,
   username_out OUT VARCHAR2,
   email_out OUT VARCHAR2,
   password_out OUT VARCHAR2,
   role_out OUT VARCHAR2
) AS
BEGIN
   SELECT username, email, password, role
   INTO username_out, email_out, password_out, role_out
   FROM USERS
   WHERE id = id_in;
END get_user;
/
/
CREATE OR REPLACE PROCEDURE get_user_by_email_password(
   email_in IN VARCHAR2,
   password_in IN VARCHAR2,
   id_out OUT NUMBER
) AS
   user_id NUMBER;
BEGIN
   SELECT id INTO user_id
   FROM USERS
   WHERE email = email_in
   AND password = password_in;

   id_out := user_id;
END get_user_by_email_password;
/
```